Journal of Virtual Worlds Research

jvwresearch.org ISSN: 1941-8477

Metaverse
Assembled 2.0

July 2011

Volume 4, Number 1
Metaverse Assembled 2.0
July 2011

Journal of Virtual Worlds Research
jvwresearch.org ISSN: 1941-8477

Volume 4, Number 1
Metaverse Assembled 2.0
July 2011

Procedural Modeling for Realistic Virtual Worlds Development

By Pedro Brandão Silva, António Coelho
FEUP/DEI, INESC Porto

Abstract:

The creation of virtual environments, corresponding to real world settings, constitutes one of the most important, but also most time and resource demanding processes in content development for virtual worlds. By querying real-world data sources, the building process can be automated, reducing the need for human intervention. However, these may present additional management difficulties due to their number, dimension, format or level of detail, therefore requiring specific techniques to operate on them. This paper describes the application of a procedural modeling solution, called PG3D System, on the creation of realistic virtual urban environments and their employ in virtual world applications. This system's implementation in spatial database management systems induces fast data access, large data manipulation features, complex query capabilities and format flexibility, allowing the fast creation of large scale virtual environments, imbued with optimized data structures, compatible with any digital game, custom modeling tool or metaverse platform.

Keywords: Virtual Urban Environments, Procedural Modeling, Shape Grammars, Spatial Engines

Procedural Modeling for Realistic Virtual Worlds Development

By Pedro Brandão Silva, António Coelho
FEUP/DEI, INESC Porto

The creation of virtual environments constitutes is one of the most demanding processes regarding the development of virtual reality applications, in the way that it requires the design of large amounts of highly detailed and quality contents. This leads to great costs and a great deal of effort, as well as long development periods. When considering metaverse platforms, where people and organizations try to introduce their real world structures, the recreation of existing urban environments has become a frequent goal. This leads to a greater immersion of the player to the scene when he recognizes it, or to an aspiration to visit the real world structures when he does not. For developers however, this presents greater difficulties, since real information must be used (Figure 1).

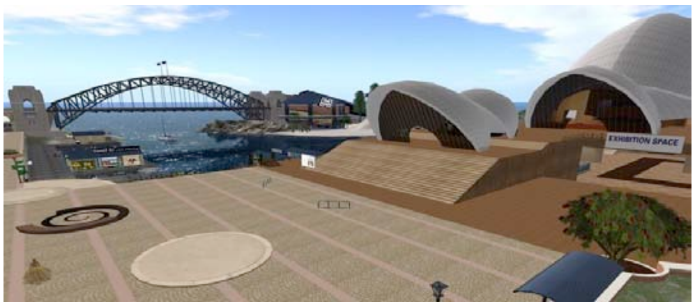

Figure 1.Sydney's representation for Second Life (Linden Research Inc., 2011)

The use of procedural methods for generating three-dimensional content for virtual reality applications has become a recurring practice, with interesting results, requiring much less effort, by generating, automatically (or at least, with little human interaction) three-dimensional models. Some guidelines and directions should be introduced by the user, or information sources that are used to describe the spaces in detail. However, sometimes the size and number of such sources is too large, thus requiring fast ways to access them and powerful machines to operate on them. On the other hand, this does not necessarily mean that such sources are detailed enough to obtain accurate models, which makes it crucial to use stochastic methods or empirical data on the urban environment to amplify the existing information. To address these issues, the PG3D modeler, a solution for procedural modeling of virtual urban environments, has been conceived.

This paper aims to describe the advantageous application of the PG3D modeler in virtual environment creation for virtual worlds. Its main characteristic is its implementation in spatial databases management systems, and operation based on a PG3D Grammar, a set of rules, which defines the instructions for the modeling procedures.

To complement the database, a management tool also exists, allowing the configuration of the necessary parameters of the modeling processes, as well as the export of generated data into a set of interoperable formats, inducing its use in various applications, such as digital games, custom modeling tools and metaverse platforms. These two parts together compose the PG3D System.

The paper is structured as follows: firstly, some related work will be presented, followed by a description of the PG3D System. Afterwards, the features of the grammar will be described, but not without explaining some important concepts first. Some information regarding the implementation will come next, followed by the results section, including the discussion. Lastly, the conclusions and some future work will be described.

Related Work

This work fits into the scope of procedural urban modeling, a research area which already possesses quite a few number of interesting approaches, some of them dedicated to specific subjects. Regarding the modeling of terrains, there is the work of Bruneton e Neyret (Bruneton & Neyret, 2008), which presents a method to integrate the various components that lie over them, such as streets, rivers and lakes, relying on a view dependent quadtree refinement scheme. For road creation, Parish and Müller used Lindenmayer Systems – or L-Systems for short (Prusinkiewicz & Lindenmayer, 1996) - originally used in the simulation of plant and organism community growth, being able to generate extensive street networks (Parish & Müller, 2001). With the same goal in mind, Chen et al. used tensor fields, which allowed interactive control over the road generation(Chen, Esch, Wonka, Müller, & Zhang, 2008). In their CityGen system, Kelly and McCabe (Kelly & McCabe, 2007) introduced a more interactive way to define primary and secondary roads.

Regarding the modeling of buildings, Wonka et al. introduced the split grammars (Wonka, Wimmer, Sillion, & Ribarsky, 2003), a new type of parametric set grammar based on the concept of shape brought up by Stiny and Gips (Stiny & Gips, 1972). He also presented an attribute matching system oriented by a control grammar, offering the flexibility required to model buildings with many different styles and designs (Wonka, et al., 2003). Based on this work, Müller et al. developed the CGA Shape (Müller, Wonka, Haegler, Ulmer, & Gool, 2006), a shape grammar capable of producing extensive architectural models with high detail. The CGA Shape is a sequential grammar (such as the Chomsky Grammar (Chomsky, 1956)), therefore all the production rules are applied in sequence, in order to allow the characterization of structure (Müller, et al., 2006). The implementation of the CGA Shape is integrated in the CityEngine framework (Procedural Inc., 2009).

Although these approaches are able to produce high quality fictitious urban environments, few are dedicated to the reproduction of real world urban environments, having therefore limited GIS data support. For this matter, Coelho presented in his work (Coelho, Bessa, Sousa, & Ferreira, 2007) the Geospatial L-Systems, an extension of parametric L-Systems which incorporates spatial awareness. This combines the ability of data amplification provided by the L-Systems (whose production rules are applied in parallel) with the geospatial systems, allowing spatial analysis of geo-referenced data. This solution is integrated into a modeling tool called XL3D modeler, which provides interoperable access to various sources of information, allowing the reproduction of real urban environments.

The PG3D Solution

The creation of realistic urban environments presents several challenges to which many authors have tried to answer over time. These include, for example, the visual correspondence in real environments, the use of information sources, or even the definition of sufficiently powerful tools able to model large urban areas. The goal of this section is to present how PG3D tries to address them. A more complete description can be found in (Silva, 2010).

The PG3D Modeler

The name PG3D is an acronym for Procedural Generation 3D, but is also related to its implementation process, namely to the technologies in which it was implemented.

The fundamental idea behind the PG3D concept is to develop the procedural modeling processes directly on a spatial database management system where either geographic data sources, as well as the created models, are saved. The modeling operations are developed as stored procedures in programming languages of the database itself. The platform that possesses PG3D modeling capabilities is called a PG3D modeler.

Since the objective lies on the modeling of real urban environments, the use of real world data is of utmost importance. Once loaded into the database, they can be queried directly without great overhead. Likewise, once started the modeling process, both final and intermediate results will be recorded in the database allowing the execution of more complex queries on them, thus expanding the range of modeling possibilities, as well as guaranteeing the safety of the generated data in case of crash or failure. Also, by relying more on database access, it can operate on smaller sets of data a time, thus overcoming the memory limits that sometimes occur for larger environments.

Use of Spatial Databases

The modeling of virtual urban environments replicating real environments leads to the need for data sources which describe the various features that compose them, such as its location, height and appearance. Such information, however, can hardly be found in a single source and organized in one place. It is essential that these sources can be connected somehow by having some common reference. In this sense it is important that the data is geo-referenced, i.e. contains a reference of its location on the earth's surface. This spatial information is fundamental to relate the many entities in the multiple sources.

The modeling of virtual urban environments replicating real environments leads to the need for data sources which describe the various features that compose them, such as its location, height and appearance. Such information, however, can hardly be found in a single source and organized in one place. It is essential that these sources can be connected somehow by having some common reference. In this sense it is important that the data is geo-referenced, i.e. contains a reference of its location on the earth's surface. This spatial information is fundamental to relate the many entities in the multiple sources.

While any database is capable of processing alphanumeric data, the spatial databases have additional features for processing spatial data types. These are normally called geometries. The Open Geospatial Consortium (OGC) specification created the "Simple Features", which defines various types of geometry (Open Geospatial Consortium, 2010). Spatial databases are also packed with multiple functions to operate over this type of data, as well as indexing features, which induce optimized search abilities.

The spatial databases are thus a solution for storage, organization, operation and management of multiple sources of geo-referenced data. Given their additional capability of editing and processing of data, format incompatibilities can easily be solved. Moreover, they can provide special features to the PG3D Modeler, since their own modeling processes act on similar data structures and may enjoy from their optimal query and geometry management abilities.

PG3D Grammar

Derived from its close relationship with the database and its implementation in own programming languages, the PG3D Grammar was born. On the one hand, it defines structures and operations, taking advantage of the PG3D concept, but on the other hand, second, it expands its range of possibilities. As the technical implementation details of this grammar do not constitute the main subject of this article, a more general overview will be presented, whereas a more complete explanation can be found in (Silva, 2010).

PG3D grammars are an extension of shape grammars (Stiny, 1980), more specifically, the CGA Shape (Müller, et al., 2006), endowed with the capabilities of geospatial awareness and relational development, stemmed from Geospatial L –Systems (Coelho, et al., 2007). Having been strongly influenced by both, it constitutes an attempt, where possible, to combine their greatest potential. Before discussing the structure of the grammars, some of its basic concepts must be explained:

PG3DShape: The PG3DShape or "shape" for short, is the main data structure manipulated by all procedural modeling processes, and therefore also in the PG3D grammar. A shape can correspond to a surface street, building, or even just a portion of it, such as a wall or corner of a window (Figure 2). A shape contains a set of geometries that graphically describe the type of element to be represented. The PG3DShape is set in a hierarchy. As will be described later, shapes evolve by successive substitution. It is therefore possible for one to know its predecessor, i.e. the shape from which it derives, allowing querying on shapes that descend from a particular predecessor.

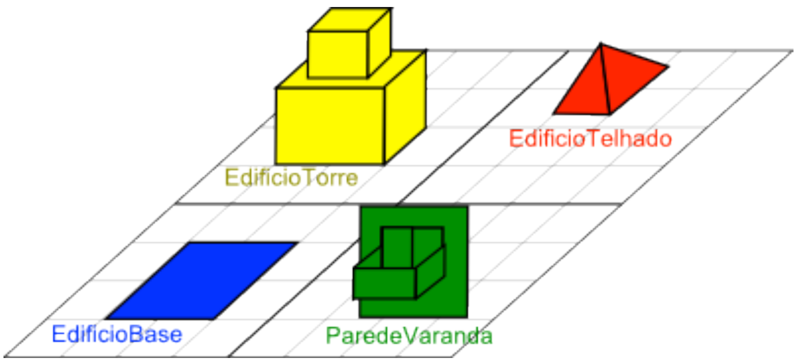

Figure 2. Various kinds of PG3DShapes

PG3DLayer: A layer is a data structure that includes one or more shapes that share the same data source, therefore a common meaning, serving as a way to organize the shapes (Figure 3). A set of PG3DLayers define the starting point for a procedural modeling process, containing instructions on data load, to be used in further steps. The geographic references contained in the layers' shapes turn their relationship possible, making it possible, such as in the example above, to combine information about the road networks and building bases with surface information, fitting into each other in an appropriate manner.

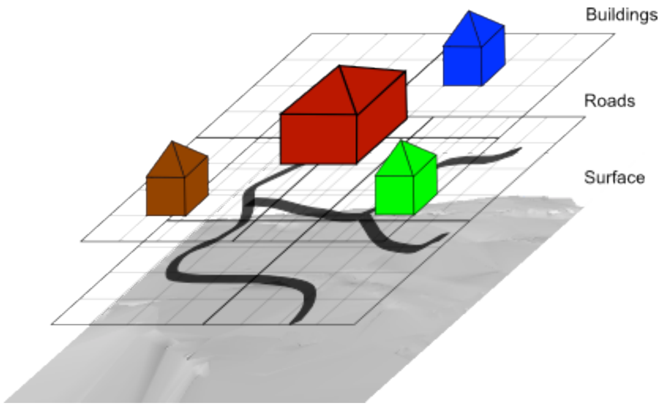

Figure 3. Relationship between multiple layers

PG3DBoundary: In complex environments such as urban areas, different construction styles and layouts can be found throughout the city (greater variations can be seen between center and suburbs). For that reason, it becomes necessary to write distinct instructions for the development of each shape depending on their location. Since the precise definition of such variations is a laborious process, it is easier to achieve that by establishing a perimeter (Figure 4). This concept, called PG3DBoundary, allows the specification of different modeling operations from one area to another.

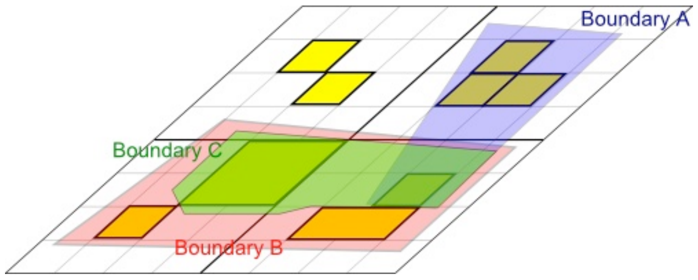

Figure 4. Multiple intersecting boundary definitions.

PG3DTag: A major difficulty that arises from handling such a large number of shapes, which may take various forms after the successive application of multiple transformations, is the management of their individual properties. It is therefore essential to create ways to select certain shape components based on their characteristics and perform operations only on these components. Although this can be partially achieved by accessing the property of each geometry, this is not always a simple task. In order to solve this problem, the concept of "tagging" components after performing modeling operations on them, was introduced (Figure 5)

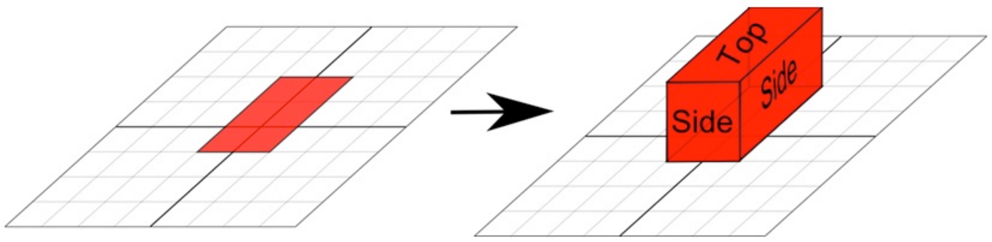

Figure 5. Example of tagged shape parts according to their creation in an extrusion. Each side of the created volume will be tagged with labels such as "side" or "top", allowing further rule definition based on these tags.

Grammar Definition

The PG3D grammar can then be described by its constitution in the following elements:

- A set of PG3DShapes, which aggregates one or more elements with graphical representation.
- The axiom, which defines the starting point of the modeling processes, defined by one or more PG3DLayers.
- A set of user-defined production rules, called PG3DRules, which delineate the modeling instructions, specifically the processes of replacement and development of PG3DShapes.

The procedural modeling process consists in successively replacing shapes in an iterative manner, starting with the axiom, and following the instructions contained in production rules. In the first iteration of the process, the layers that are part of the axiom (which indicate what sources of data should be loaded) are analyzed. After this point, having already a set of shapes, the procedure is developed iteratively - at each iteration, for each shape created, a production rule matching that shape is searched and applied. Once this is done for all shapes, it moves on to the next iteration which will handle the newly created shapes in the previous iteration. This process is repeated until no more rules are applicable to any shape or if an iteration limit, imposed by the user, has been reached.

The main idea behind the successive replacement of shapes is of progressive creation of detail, which means the models are incrementally improved. Thus it is possible to generate simple, usable models in a first step, and evolve them according to available data sources and rules. The result of each step can be queried separately, which can be extremely useful in virtual world applications to generate various levels of detail of the same environment.

Production Rules

Production rules are structured in the following form:

Predecessor → Successor

In its most basic form, the definition of the predecessor consists of a capitalized symbol composed of alphanumeric characters. This rule shall be applied to all shapes that have a matching symbol. The successor, in turn, can become quite complex, containing a sequence of modeling operations and symbols, which will mark each new shape creation. Each new shape will be the result of applying all previously called operations on the predecessor shape, and will act as its replacement. Consider the following example:

Shape1 → translate(vector3(0,10,20)) **Shape2** scale(vector3(5,5,5)) **Shape3** colorShape(rgba(255,0,0,255) **Shape4**;

This rule will replace all shapes called "Shape1" by 3 different shapes. "Shape2" will correspond to a simple translation of "Shape1", "Shape3" will be 5-time bigger version of "Shape2", and "Shape4" will be equal to "Shape3", except red colored. The operations are therefore applied in sequence, while the shape naming "saves" these changes as new shapes, which will replace "Shape1". Rule definitions may contain additional features to allow the construction of more complex structures (Silva, 2010).

Parametric Rules: From one rule application to the next, it may be necessary to send certain information. For that reason, the production rules do accept parameter definition, in a similar way functions in programming language like C++ do.

Conditional Rules: The definition of rule conditions is essential for the development of shapes, especially when loaded with external information sources or when subjected to stochastic processes. The PG3D Grammar thus includes support for control structures, such as the IF…THEN…ELSE expressions.

Stochastic Rules: To compensate for the lack of information that some sources may contain, the use of randomness can become a form of data amplification, when used and controlled effectively. The parameterization can be achieved through specific expressions provided which operate based on probabilities, which can control, for instance, the frequency of appearance of a particular characteristic in a certain environment.

Parenthetic Rules: By using a stack it is possible to save and load "states", i.e. results of transformations over shapes. By using PUSH, the current state is saved on a stack, and loaded from it when POP is used. The use of the name "parenthetic" derives from its concept from L-Systems, which use the "[" and "]" operators to achieve this goal, but to avoid conflicts with array definitions, it has been replaced in PG3D.

Attributes, Limits and Textures: Using the same values in more than one production rule is a common practice. Therefore it is possible to declare constants only once, and use them multiple time in rules.

Transforming, Reading and Accessing Real Data and Geometries

The first step in using geographical information sources is to understand their format and how to access them. PG3D considers GIS Data which follows the "Simple Features" specification (Open Geospatial Consortium, 2010; Open Geospatial Consortium Inc., 2006) by the Open Geospatial Consortium, meaning that it stores primitive geometrical data types, such as points, lines and polygons. In the process of reading data, only two fields are loaded: the geometry and a primary key that identifies the record uniquely. Its existence allows, on one hand, to operate always on geometries (which must have spatial information) and, secondly, to own a reference to each record that was read. This allows the query of other fields of the record (text information, other numeric values, etc.).

Conditional and Characteristic Development

One of the major feature of the PG3D Grammar is its ability to conditionally develop each shape (or any of its parts, namely vertices or polygons) based on its properties. In other words, the evolution of a shape may depend on its current status or on the corresponding data that may exist in the data sources. The control structure IF...THEN...ELSE is one way of checking these properties. The other exists at function level, in that the some support a boolean parameter, allowing the operations to be applied only to parts of a shape fitting a certain condition. To indicate the intent to act based on the state of the shape or its vertices or polygons, PG3D supports the three reference symbols starting by the percentage sign, **%s**, **%p**, **%v**, respectively.

Geospatial Awareness

The concept of contextualization in geospatial PG3D grammars derived from its application in Geospatial L-Systems (Coelho, et al., 2007). Its main idea consists in developing shapes not independently, but based on their surroundings, avoiding the creation of unrealistic structures. This is especially important when performing occlusion tests.

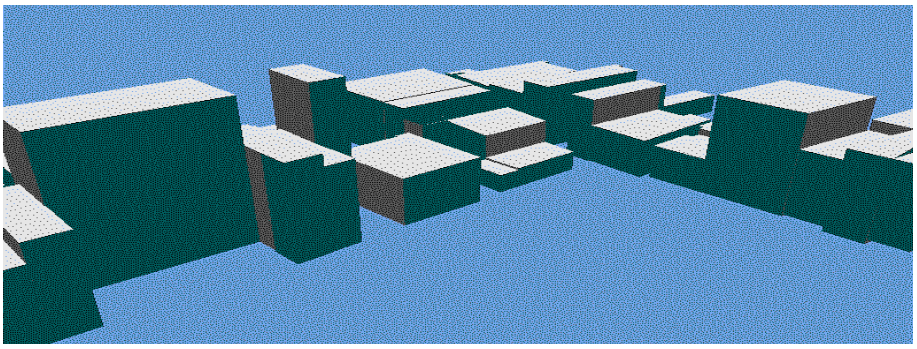
Figure 6. Colored façades demonstrate geospatial awareness

Supposing a couple of buildings share common walls: these should not contain any windows, balconies or front doors. Since all the elements are spatially referenced, such check is easily achieved. Figure 6 shows building walls which do not have any other wall in front of them at a distance lesser than 3 meters away. This makes them potential good façades for buildings.

PG3D Graphical Interpretation for Real-Time Virtual Applications

Another important feature of the PG3D System lies on its graphic interpretation of the modeled shapes. As mentioned before, PG3DShapes are built from geometries that describe their appearance. To be able to view them in any graphical tool or framework, it is necessary that its structure is previously "translated". However, this operation can become complicated depending on the requirements and potential of the target platform.

PG3D Structure as a Key Feature

PG3D operates with the most basic forms of graphical data on which the computer graphics normally operate: vertices and their conjunction in polygonal meshes, especially in triangular meshes. This structure is thus easily integrated with development platforms, graphic editing or viewing, but especially game or virtual world engines.

This type of data structure, integrated with capabilities of complex queries on the database, induces increased optimization capabilities. For example, the ability to detect vertices with the same characteristics in large sets of shapes can dramatically reduce the amount of data to be drawn by the graphic card when rendering the entire scene. This type of approach, tempered by some optimization techniques available nowadays, such as vertex buffers and index buffers can allow the visualization of very large urban environments (Figure 7), even in real time applications such as metaverse platforms.

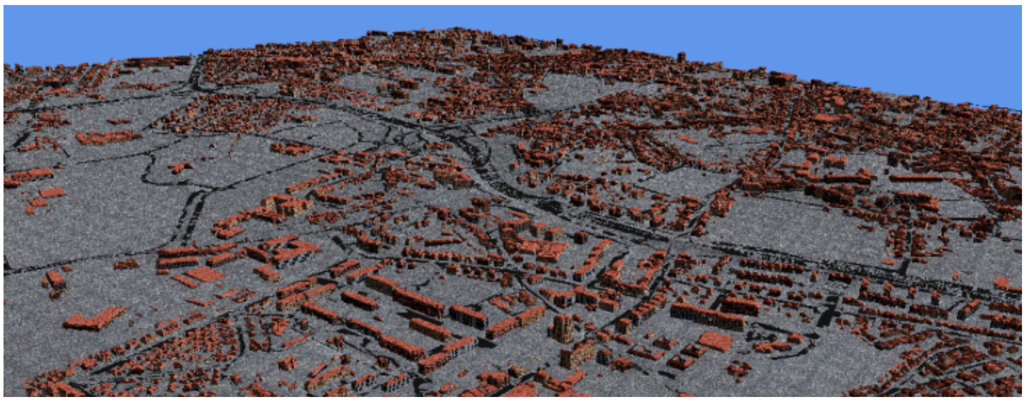
Figure 7. 10km2 urban area, featured by PG3D

Data Visualization and Export

The implementation of the PG3D modeler in databases makes it difficult, if not impossible, to display graphic information directly in the database management system tools. As such, the use of the generated data has to be performed by external applications, such as services or clients that access the database for extraction and subsequent visualization or export of the generated models. Together with the PG3D modeler, they form the PG3D System.

Implementation and Technologies

PG3D's full name refers, besides to the reference to its objective, to the technology in which this first prototype was implemented. PG3D is therefore an acronym for the triple **P**ost**g**reSQL, **P**ost**G**IS, **P**rocedural **G**eneration **3D**.

The database management system chosen was PostgreSQL, using the PostGIS spatial extension, which allows not only the load and storage of geographic information sources, but also the execution of spatial operations in the created data structures. The chosen programming language was PL/PgSQL, which takes advantage of its native implementation in the database to increase the power, flexibility and performance of the created functions. Since this database management system does support neither 3D visualization nor custom file writing capabilities to export the data, a client application was conceived (Figure 8). This makes use of.NET framework and C# to provide easy and fast ways to manage, visualize and export the produced environments.

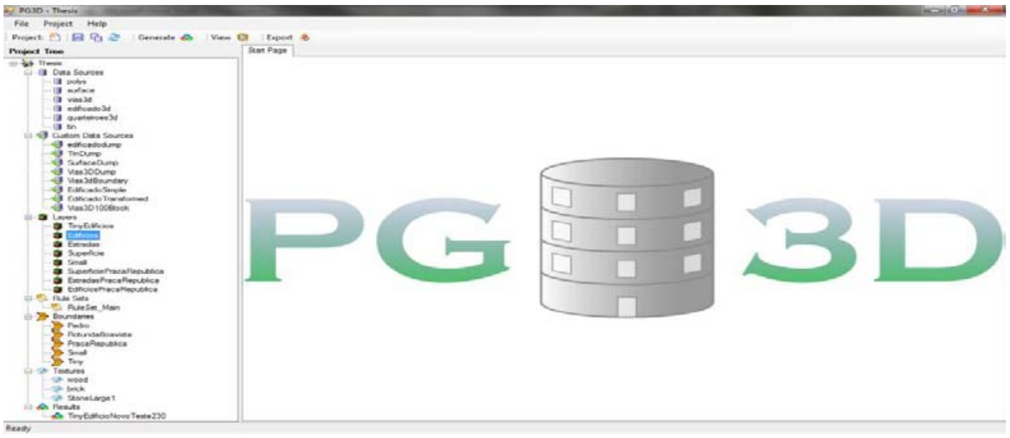

Figure 8. PG3D Client Interface

Results

The PG3D System was developed in order to facilitate the modeling of virtual urban environments by reducing the human interaction in the process, thus contributing to less effort, less design time and hence lower production costs. It is therefore ideal for applications that make use of this type of content such as digital games and virtual world applications. To demonstrate the advantages stated by the PG3D concept, some tests were performed at various levels.

Capacity, Quality and Level of Detail

An important factor in modeling is the quality and level of detail of the created elements. To be able to achieve certain levels of detail, it is necessary that the modeling processes are capable to reproduce existing buildings (or at least most of them) with enough precision. The development of this feature was a major goal of the PG3D modeler. An example of a created building is displayed in Figure 9.

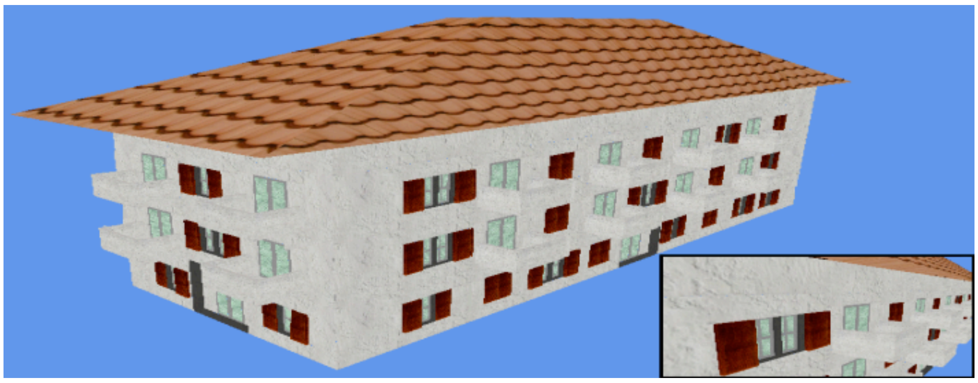
Figure 9. Model of a house, featuring high detailed windows, shutters and balconies

The presented model shows that the various elements of the building façade of a house can be defined with great detail. The design of such structures, however, needs the definition of a larger number of production rules, which, in turn, take more time to process.

The application of production rules is done in an iterative way, which allows a progressive improvement of the models' detail. Due to the possibility to load intermediate states of development of each model, multiple levels of detail of the same environments can be used. This is especially useful in virtual world applications, in order to adapt the displayed detail to the machine's capabilities, making the world navigable even in slower ones.

Visual Resemblance and Accuracy

In order for a virtual world visitor to be able to recognize the created virtual urban environment, is it important that its elements hold as many similarities to the real ones as possible. This is achieved mostly through the distribution of roads and buildings, but also through a lot by smaller details, such as façade outline and contained features. Due to the limited amount of data sources used until this moment, only the first perception has been achieved (Figure 10), so some work in this area is still being developed.

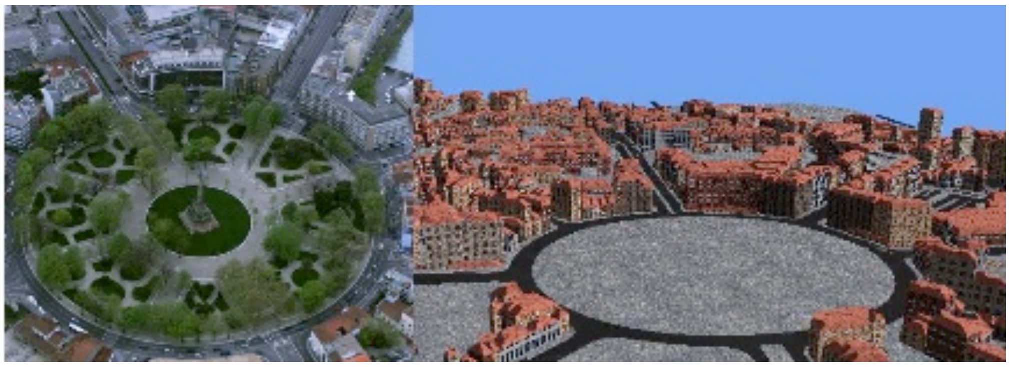
Figure 10. Comparison of the Boavista Roundabout in Porto, Portugal and its correspondent virtual environment

Definition of modeling processes

The definition of production rules represents the main task to be done by the user intending to obtain procedurally created models. However, such may not always be an easy task, since both data sources and projected models must be carefully analyzed beforehand. Even so, when regarding more basic environments, such as the simple definition of building blocks and streets standing on a surface (Figure 9), the rules can be very straightforward and simple. For the following example, three layers corresponding to the surface, road and building, are considered, which contain real world spatial information (Figure 11).

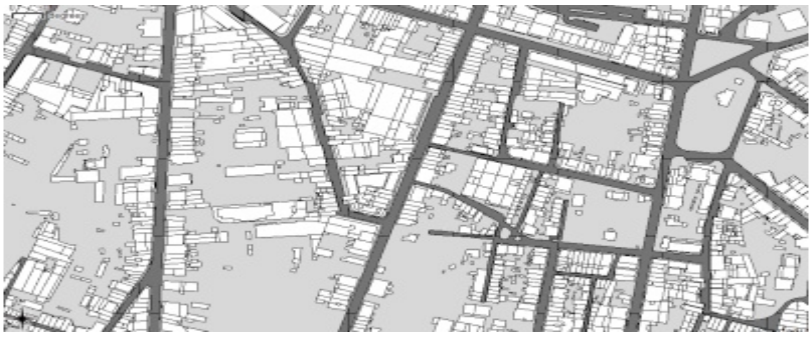
Figure 11. Surface, city blocks (light grey), street (dark grey) and building information (white) viewed using GIS software.

The rules that follow have been labeled with letters in order to help their explanation:

```
Surface →  surfaceOptimize(20,20)  //a
           setTexture(@t_Stone1)    //b
            SurfaceStone;          //c
```

The rule above takes the information about the surface (contained in a shape called "Surface"), optimizes its data structure (a) and applies a texture (b), being its path saved on a variable named "Stone1". In the end, the result is saved as a new shape called "SurfaceStone (c).

```
Roads → surfaceOptimize(20,20)       //a
     setTexture(@t_Asphalt)       //b
     placeVertexOnCustomSurface('Surface') //c
                  Roads;
```

This rule operates over the information of the roads, applies an asphalt texture (b) after optimizing it (a), and places the roads on the surface, adapting it to the elevation features (c).

```
BuildingFootprint → placePolygonOnCustomSurface('SurfaceDump')  //a
                extrude(double(record(%p),'height')) //b
                  STOC  15% setTexture(@t_Window1) //c
                            15% setTexture(@t_Window2)
                            15% setTexture(@t_Window3)
                            15% setTexture(@t_Window4)
                            15% setTexture(@t_Window5)
                      25% setTexture(@t_Window6) ENDSTOC
                   setUV(5,5)                     //d
                       {hasTag(%p,'Edificio.Top')
        %each : BuildingTop, %rest : BuildingSides};    //e
```

In this step, the building footprints are loaded and placed on the surface, like the roads (a). The information of their heights, also contained on the database, is used to extrude the footprints (b), creating volumes of the buildings with real heights. To introduce variety in their textures, a stochastic method is used to choose them, based on probabilities (c). This way, each texture (except the last) has a 15% probability to be chosen. Afterwards, the UV mapping of the texture is applied based on a fixed size (d). Lastly, the shapes are divided (e): the building sides are saved in a shape called "BuildingSides" and the top face is saved in a shape called "BuildingTop". This new shape is considered in the rule below, which creates a roof-like structure (a) from this top face and applies a red brick texture (b).

```
BuildingTop → extrudeTaper(3,5) //a
                 setTexture(@t_RoofRed) //b
                        BuildingRoof;
```

The following rules, when applied, produce the example portrayed in Figure 12:

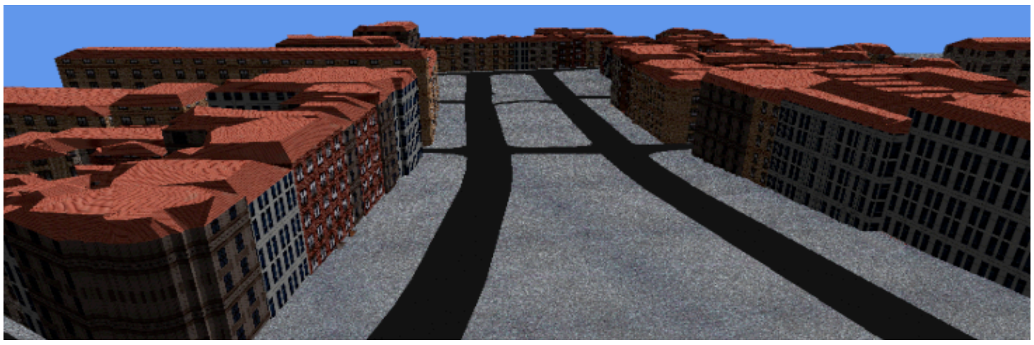
Figure 12. Aliados Avenue in Porto, Portugal

As it can be seen, even through a small number of production rules, very satisfactory results can be obtained. These simple rules can be applied to as many models as intended in a matter of seconds, a task that could take hours or days when modeled individually with manual tools.

Integration with Virtual World Applications

The development of quick and simple ways to put the created models into practice constitutes one of PG3D main concerns, and such is achieved, as mentioned, by the visualization and export abilities of the PG3D client application. The software itself uses the XNA Framework (Microsoft Corporation, 2011) (Figure 6, Figure 7, Figure 9, Figure 10 and Figure 12), a popular game development framework, to display the tridimensional models. Its implementation in high-level languages and powerful features has made XNA quite attractive for a growing number of users, and therefore a recurrent tool for the creation of virtual worlds. Although its employment in game creation is mostly single player oriented, some attempts are being made (Walsh, 2009) regarding massive multiplayer online games (MMORPG).

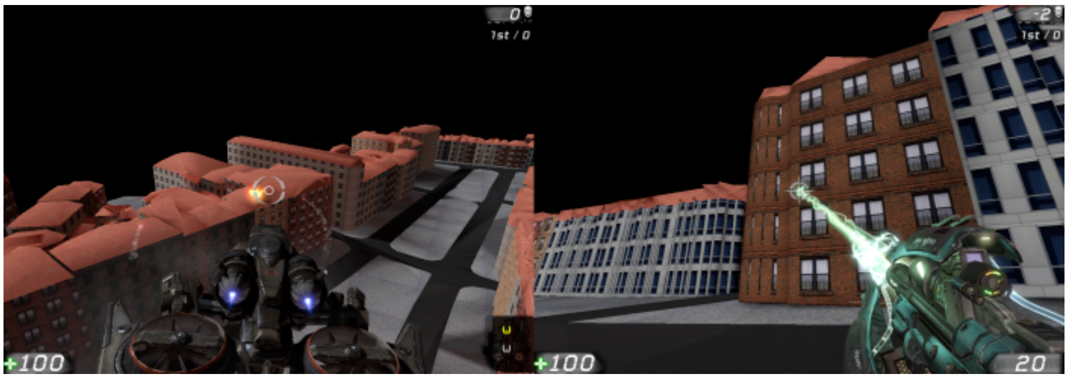

Figure 13. Aliados Avenue in Porto, Portugal, represented in a virtual environment played on Unreal Tournament 3 (Epic Games, 2009)

In this sense, their direct integration with the PG3D System could enjoy the additional possibilities of already implemented runtime model loading and large-scale environment support.

In order to employ the obtained models in further platforms, PG3D supports their export to COLLADA (Khronos Group, 2011), a popular exchange format for 3D content. Figure 13 shows the use of a PG3D created environment in “Unreal Tournament 3”, an online multiplayer game. The import procedure consisted simply on loading the COLLADA file using the powerful Unreal Development Kit and defining some properties.

For other applications, where a more restricted number of formats is accepted for their content input, the data import cannot always be done directly, being an intermediate format change needed. The popularity of COLLADA has led to a significant support by various authoring tools. This means, that it is not only possible to import and export to various different formats, but to carry manual changes on the procedurally created models, if desired, in order to refine, correct or add further details (Figure 14).

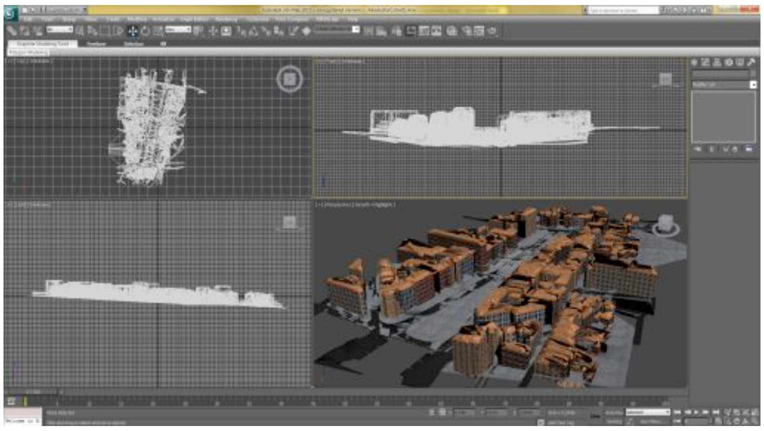
Figure 14. Generated urban model processed by the 3Ds Max (Autodesk, 2009) authoring tool

As a further example of a virtual world application, the same models were tested on Open Cobalt (Duke University, 2010), which makes use of the Open Croquet SDK to manage the virtual environments on a large user scale.

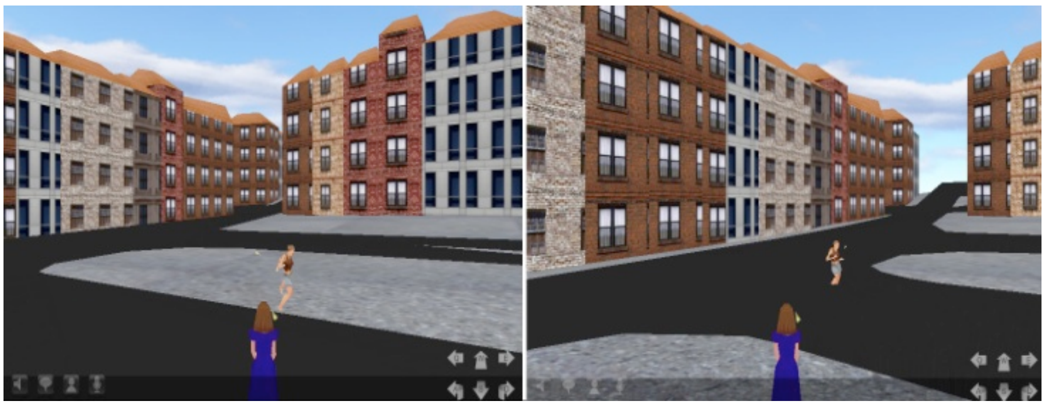
Figure 15. Open Cobalt's "Alice" and "Dancing Girl" avatars in a virtual representation of Porto's downtown

Open Cobalt intends to be a platform for collaborative user virtual workspaces, in which the users can easily add new contents and manipulate them in real-time. It is able to import external 3D content in the ASE format, which is a standard ASCII format common to many authoring tools. After converting to this format, a single drag and drop action of the file to the display window is sufficient to import the PG3D generated models into this virtual environment application (Figure 15).

Conclusions and Future Work

The PG3D System is still in an alpha phase, but shows enormous potential regarding the procedural modeling for development of virtual representations of real-world urban environments. In short, the PG3D System features:

- Loading and organization of geo-referenced information sources regarding real-world urban spaces;
- Procedural modeling of tridimensional virtual environments, oriented by user-defined production rules, which operate over these sources;
- Visualization and export of the generated models for use in multiple games, authoring tools and virtual world applications.

Besides presenting itself as a much faster, simpler and cost-effective method of content creation, the results are likely far more accurate and consistent than if they were created manually. The laborious task of mass modeling is, in such cases, much easily led by procedural methods.

On a more technical view, the PG3D concept has demonstrated to induce the following advantages, comparing to other procedural approaches, such as:

- Quick access to data sources, since it reduces the data access overhead;
- Greater modeling limits, since it can operate over small sets of data at the time, therefore is not limited by environment size;
- Greater safety, since it stores every step on disk and can be recovered even in the case of crash;
- Easy data source creation and manipulation, due to the availability of database query languages, such as SQL;
- Easy execution of complex operations through custom queries, derived from the relational database structures and query languages. An example would be the creation of optimized models formats for use in real-time viewing applications.
- Easy integration with any platform, since the modeling functions are stored in a database and can be accessed by any client or service that intends to enjoy from their procedural modeling capabilities.

Despite these advantages, the results still have to be further developed, especially to achieve greater visual resemblance with existing environments. The achievable model quality is high, though their correspondent creation time to attain them is still too elevated for more detailed cases. Although the time to manage the sources is reduced, the modeling operations are still too slow, therefore optimizations still have to be done. The usage of the generated data has been successful, but in order to integrate them more easily with any game or metaverse platform, additional formats should be supported in the future.

Bibliography

Bruneton, E., & Neyret, F. (2008). Real-time rendering and editing of vector-based terrains. *Computer Graphics Forum, 27*(Eurographics 2008), 311-320.

Chen, G., Esch, G., Wonka, P., Müller, P., & Zhang, E. (2008). *Interactive procedural street modeling*. Paper presented at the ACM SIGGRAPH 2008 papers, Los Angeles, California.

Chomsky, N. (1956). Three Models for the Description of Language. (IRE Trans. Information Theory (2),), 113–124.

Coelho, A., Bessa, M., Sousa, A. A., & Ferreira, F. N. (2007). Expeditious Modelling of Virtual Urban Environments with Geospatial L-systems. *Computer Graphics Forum, 26*(4), 769-782.

Duke University. (2010). Open Cobalt Retrieved 26/02/2011, from http://www.opencobalt.org/

Kelly, G., & McCabe, H. (2007). Citygen: An Interactive System for Procedural City Generation. (GDTW '07).

Khronos Group. (2011). COLLADA - Digital Asset and FX Exchange Schema Retrieved 26/2/2011, from http://www.khronos.org/collada/

Microsoft Corporation. (2011). App Hub Retrieved 26/2/2011, from www.xna.com

Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Gool, L. V. (2006). *Procedural Modeling of Buildings*. Paper presented at the ACM SIGGRAPH 2006 Papers, Boston, Massachusetts.

Parish, Y. I. H., & Müller, P. (2001). Procedural Modeling of Cities. (SIGGRAPH 2001), 301–308.

Procedural Inc. (2009). 3D Modelling Software for Urban Environments. *Procedural* Retrieved 26/2/2011, from http://www.procedural.com/

Prusinkiewicz, P., & Lindenmayer, A. (1996). *The Algorithmic Beauty of Plants*: Springer-Verlag.

Silva, P. B. (2010). *Modelação Procedimental para Desenvolvimento de Jogos de Computador.* Master Master Thesis,, University of Porto, Porto.

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B, 7*(3), 343-351.

Stiny, G., & Gips, J. (1972). *Shape Grammars and the Generative Specification of Painting and Sculpture.* Paper presented at the Information Processing '71.

Walsh, J. (2009). Level-Grind Online - Programming an MMORPG in C# with XNA Retrieved 26/2/2011, from http://levelgrindonline.com/

Wonka, P., Wimmer, M., Sillion, F., & Ribarsky, W. (2003). *Instant architecture*. Paper presented at the ACM SIGGRAPH 2003 Papers, San Diego, California.